Part 3 : The HTML

The HTML is the means to display the image on a browser. The general format of just about any HTML file is as follows

```
<html>
<head>
</head>
<body>
</body>
</html>
```

Within the body of the HTML page we want to construct a 'table' which looks remarkably like the one we've drawn.

The image table

A table in HTML lies between <table> and </table> constructs. A series of rows may be specified using <tr> and </tr> (tr stands for 'table row'). Within each row are table data elements (<td> and </td>) which is where we will put each of our fragmented images. Each of these HTML commands can have 'attributes' which specify certain characteristics. Fortunately we don't need many, and the shape of the table is determined by the image sizes themselves. So the table declaration for the entire image we have:

```
<table align=center border=0 cellpadding=0 cellspacing=0>
</table>
```

The border, cellpadding and cellspacing attributes just specify that there should be no gaps between the data elements. Within the table we lay down the rows, however many there are (3 in our example). So the first 'top' row is:

```
<tr>
  <td colspan=10><img src="images/top.jpg"></td> </tr>
</tr>
```

The <td> command has a 'colspan' attribute. This says that the element spans 10 columns (i.e. all of them). This is because this is a static segment and is as wide as the entire image. Between the <td> and </td> commands is an <img> command. This is the command which loads in the image we want in this data element of the table. It too has attributes. The src attribute gives the file name of the image. In the example it is a relative reference, but it could also be a URL. All the static images are specified this way.

The dynamic images are a little more complex

```
<tr>
  <td><img name="dl" src="images/display_left.jpg"></td>
  .
  .
  <td><img name="d4" src="images/lcdoff.jpg"></td>
  <td><img name="d3" src="images/lcdoff.jpg"></td>
  .
  .
  <td><img name="dr" src="images/display_right.jpg"></td>
</tr>
```

Here all ten columns of the table have a data element. The left and right display images mark the ends, with the other 8 elements loaded with a default image (in this example 'off' images). An important new attribute is added here: 'name'. Every dynamic image command must have a name so that it may be referenced by the javascript. This is how the images can be changed in response to input.

Any switches are done in a similar way, but it is extremely unlikely that your switches exactly line up with the boundaries of your display. So the row is initially treated as static (a single spanning all columns), and then between the single <td></td> pair a new table is defined. This table is as wide as the main table, has a single row, and then as many data elements as are required for the switch row, with dimension attributes to suit.

Having done all this for all the table elements, and with the table within the body of an HTML file you should be able to open it with a browser and see your composite image displayed. If there are any gaps or missing segments or distorted images, go through and check your dimension numbers and file name references.

Switch and Key maps

To get input from the user, mouse clicks or keyboard entry needs to be detected. For the mouse we will need to define areas on the images which correspond to keys or switches on the calculator. This is done with a <map> command.

```
<map name="keymap">
  <area shape=rect coords=" 16, 39, 46, 68" href="javascript:key_pressed('o')" title="o"></area>
  .
  .
  <area shape=rect coords="169,181,201,207" href="javascript:key_pressed('=')" title="="></area>
```

```
</map>
```

In the above fragment a map called keymap is defined. In it an area is defined with a rectangular shape and a reference to a javascript function 'key_pressed' with a single argument indicating the calculator key (more on javascript later). The coordinates specified in the area command are relative to an image, where the origin starts at the top left of the image. We associate this map with an image in the table as shown in an example below:

```
<tr><td colspan=10><img src="images/keys.jpg" border=0 usemap="#keymap"></td></tr>
```

The usemap attribute of the <img> command connects to the map, and whenever the mouse hovers over the area of the image specified in the map the cursor changes from a pointer to a 'press' type shape. If the mouse key is pressed then the referenced javascript functions is called. As a bonus, the map's title attribute allows the display of a message when the mouse is hovering over the area. In this case it is just the keyboard equivalent of the mouse selection. The mapping is repeated for all input regions (keys and switches etc.). If an image has multiple input areas (e.g all the keys) multiple area commands are specified within a single map and only one map associated with the image.

Below is an image constructed in this way. Try hovering the mouse over the various keys. If you have a status bar showing in your browser, note what it indicates when you're over a key.

If we want to have keyboard input as well as mouse selection, then another attribute is defined; this time for the HTML body command:

```
<body topmargin=0 leftmargin=0 marginheight=0 marginwidth=0 onkeypress="javascript:keyboard(event)">
```

The onkeypress attribute specifies a javascript function to call whenever a keyboard key is pressed---in this case 'keyboard'. An argument 'event' is passed to the function so that information about which key has been pressed can be determined. The other body attributes remove borders etc., much like in the table. This is so that we can (should we wish) place the whole page in a popup window, without a border being placed in it. The good news is that the main HTML is done! The bad news is that the hard part is next! (Not really). The full HTML for the example (minus the javascript) can be viewed here.

"How to write a calculator simulator"
<- Prev Page Next Page->

www.anita-simulators.org.uk/calc/calc_example/article2_3.htm 2/2